

Process of Isolating The Emission Component in the GMI Code

JULES KOUATCHOU
NASA Goddard Space Flight Center
SIVO/ASTG - Code 610.3
Greenbelt, MD 20771
Jules.Kouatchou.1@gsfc.nasa.gov

March 10, 2006

Abstract

In the Fall 2005, we started isolating the major components of the GMI code with the objective of making the code more flexible, more accessible, modular and ESMF complaint. In this report, we describe how the isolation of the Emission component was accomplished.

Contents

1	Introduction	3
2	Legacy Emission Control Routine	4
3	Removing “Include” Files	4
4	Variable Intent and Ownership	5
5	Derived Types	5
6	Run Routine for the Emission Component	5
A	Legacy Code: Emission Control Routine	6
B	Legacy Code: Variables used in the Emission Operator	6
C	Emission Derived Type and Member Variable Manipulations	8
D	Code for the RunEmission Routine	9

1 Introduction

The GMI model is a modular chemistry-transport model (CTM) with the ability to carry out multi-year assessment simulations as well as incorporate different modules, such as meteorological fields, chemical mechanisms, numerical methods, and other modules representing the different approaches of current models.

The initial design of GMI offered the possibility of adding and testing new components without major effort. As the code evolved, it became less flexible. Individual components became dependent of each other in the sense that variables were passed from one component to another through Fortran common blocks but not as arguments of routines.

In the current version of the code, it is difficult to determine which module owns (declare, allocate, initialize, update) a given variable. Variables are declared, allocated and initialized in groups which members belong to different components. In addition, all namelist variables are read in at once by a unique routine.

We want, by componentizing the GMI code, to

1. Identify the major components (Chemistry, Emission, Advection, Convection, Deposition, Diffusion, and Diagnostics)
2. Isolate each component by removing within it include files that are not part of the component.
3. Make the code more flexible.
4. Make the code ESMF complaint.

To accomplish it, we carry out the following steps:

- For an existing component, pass all the variables needed by it as arguments of its control routine. This process involves removing all the common blocks "external" to the component. The "internal" ones only include variables not seen outside the component.
- For each variable in the code, determine its ownership.
- For each component, create a derived type which member variables are own by the component.
- For each member variable, identify how it is initialized and/or updated and which component(s) use it.
- For each member variable, as part of derived type created, write (if necessary) routines to `Allocate` it, to `Set` its value and to `Get` its value.
- Write wrapper routines which argument are derived types only.

At the end of the process

- The declaration of variables, their allocation and initialization will be the responsibility of components.
- A namelist variable will be set by the component it belongs to.

- Each public member variable of a given component will be accessible anywhere in the code through a call of the corresponding `Set` or `Get` routine.

Three main routines are then written to refer to a given component:

Initialize: is called once to read namelist variables, do initial setting and perform variable allocation.

Run: is used to invoke the component for model time stepping.

Finalize: is called if necessary to deallocate variables.

The objective of this report is to provide an example on how the process of componentization was carried out in the GMI code. We choose the GMI Emission operator and explain how each of the step applies to this case.

2 Legacy Emission Control Routine

In the GMI legacy code, the code for the Emission control routine is in the file `emiss_update.F90`. The top part of the control routine (*Update_Emiss*) is presented in Appendix A. We note that the routine contains three GMI include files that are not part of the Emission operator.

3 Removing “Include” Files

We want to remove the include files `gmi_dims.h`, `gmi_control.h` and `gmi_gen_io.h`. To do that, need to identify the variables in these include files that are used in *Update_Emiss*. They are:

```
i1, i2, ju1, j2, k1, k2
ilo, ihi, jul0, jhi
i1_g1, i2_g1, ju1_g1, j2_g1
num_species, mw
iisoprene_num, ico_num, ino_num, ipropene_num
emiss_timpqr, ndust, naero, nst_dust, nt_dus
num_emiss, emiss_opt, emiss_dust_opt, do_gcr
do_aerocom, do_semiss_inchem
num_time_steps, ndt, tdt, nymd
do_drydep
met_opt
pr_surf_emiss, pr_emiss_3d
```

The above variables will now be passed as arguments of *Update_Emiss*.

We should also note that *Update_Emiss* calls the routines *Read_Emiss_Data1*, *Read_Emiss_Data2*, *Add_Emiss_Llnl*, *Update_Emiss_Harvard*, *Add_Emiss_Harvard*, *Split_Time_Flds*, and *Add_Emiss_Gsfc*. Those routines may also contains GMI include files (we do not touch at all include files that are part of the Emission component) that should be removed too. We again need to list the variables belonging to the include files that are used in the routines. We come up with the additional variables

```
veg_infile_name, lai_infile_name, light_infile_name, isopconv_infile_name,
monotconv_infile_name, fertscal_infile_name, precip_infile_name, soil_infile_name
emiss_aero_opt, emiss_dust_opt, emiss_map, emiss_map_dust,
emiss_map_aero
```

```
pt
chem_opt
trans_opt
```

They too should be arguments of *Update_Emiss*.

4 Variable Intent and Ownership

Giving the new set of arguments of *Update_Emiss*, we want to give the intent and ownership of each variable in the set. Appendix B lists all the variables, their description, intent and ownership.

We note that some emission related variables are just input (IN) to *Update_Emiss* because they are namelist variables or are read in or set before the emission control routine is called. In the new design, these variables will set in the Emission Initialize routine.

5 Derived Types

Now we describe how the Emission derived type was created. We went through the entire code and gather all the variables that are owned by the Emission operator. The variables now become members of the derived type called *Emission*. We need to write basic routines to allocate, set and get each variable. Appendix C shows how it is done..

The following derived types were created to classify and manipulate GMI variables:

Chemistry: Contains chemistry related variables

Convection: Contains convection related variables

Deposition: Contains deposition related variables

Diffusion: Contains diffusion related variables

Emission: Contains emission related variables

GMI_Dimensions: Contains domain decomposition variables.

MassPressure: Contains the variables *ai*, *bi*, *am*, *bm*, *mass*, *mcor*, *dlatr*, *latdeg*, and *londeg*.

MetFields: Contains the list of met fields.

ModelStepping: Contains the variables *pt*, *gmi_sec*, *tdt*, *ndt*, *num_time_steps*, *nymd*, *nhms*, *start_ymd*, and *leap-year-flag*.

SpeciesConcentration: Contains the species concentration array *const*.

For each derived type, whenever necessary we wrote "Allocate", "Get" and "Set" routines to manipulate its member variables.

6 Run Routine for the Emission Component

In Appendix D, we provide the entire code for the *RunEmission* routine that is now the new wrapper to call the Emission operator in the GMI time stepping routine.. To explain how the routine was written, we want to make the following comments:

- The arguments of the new routines are mainly derived types.
- Basic information on species (number of species, species indices, molecular weight) are now passed through modules.
- The variables (almost all of them) that were arguments of the legacy code routine *Update_Emiss* became local variables. The ones in *Update_Emiss* with intent IN or/and INOUT need to be set using derived types before doing any computation. Those in *Update_Emiss* with intent INOUT or/and OUT need to be passed to the derived types before exiting the new routine.

A Legacy Code: Emission Control Routine

```

subroutine Update_Emiss
& (lwi_flags, latdeg, londeg, mc当地, emiss_isop, emiss_monot, &
& emiss_nox, radswg, surf_air_temp, surf_rough, con_precip, &
& tot_precip, ustar, mass, max_cloud, ran_cloud, kel,
& surf_emiss_out, surf_emiss_out2, emiss_3d_out, &
& const, emiss, emiss_dust_t, emiss_dust, &
& emiss_aero_t, emiss_aero,pbl,ai,bi,humidity,pctm1, &
& cldmas0, press3c, press3e, cmi_flags, dtrn, flashrate, &
& lightning_no)

implicit none

# include "gmi_dims.h"
# include "gmi_control.h"
# include "gmi_gen_io.h"

! -----
! Argument declarations.
! -----

integer :: lwi_flags      (ii:i2, ju1:j2)
real*8  :: latdeg        (ju1:g1:j2_g1)
real*8  :: londeg        (ii:g1:i2_g1)
real*8  :: mc当地        (ii:i2, ju1:j2)
real*8  :: emiss_isop     (ii:i2, ju1:j2)
real*8  :: emiss_monot    (ii:i2, ju1:j2)
real*8  :: emiss_nox      (ii:i2, ju1:j2)
real*8  :: radswg         (ii:i2, ju1:j2)
real*8  :: surf_air_temp  (ii:i2, ju1:j2)
real*8  :: surf_rough     (ii:i2, ju1:j2)
real*8  :: con_precip     (ii:i2, ju1:j2)
real*8  :: tot_precip     (ii:i2, ju1:j2)
real*8  :: ustar          (ii:i2, ju1:j2)
real*8  :: mass           (ii:i2, ju1:j2, k1:k2)
real*8  :: max_cloud      (ii:i2, ju1:j2, k1:k2)
real*8  :: ran_cloud      (ii:i2, ju1:j2, k1:k2)
real*8  :: kel             (ilo:ih1, julo:jhi, k1:k2)
real*8  :: surf_emiss_out (ii:i2, ju1:j2, num_species)
real*8  :: surf_emiss_out2(ii:i2, ju1:j2, 4)
real*8  :: emiss_3d_out    (ii:i2, ju1:j2, k1:k2, num_species)
real*8  :: const           (ii:i2, ju1:j2, k1:k2, num_species)
real*8  :: emiss           (ii:i2, ju1:j2, k1:k2, num_emiss, emiss_timpqr)
real*8  :: emiss_dust_t   (ii:i2, ju1:j2, ndust, nst_dust:nst_dust+nt_dust-1)
real*8  :: emiss_dust     (ii:i2, ju1:j2, ndust)
real*8  :: emiss_aero      (ii:i2, ju1:j2, naero)
real*8  :: emiss_aero_t    (ii:i2, ju1:j2, naero, emiss_timpqr)
real*8  :: pbl             (ii:i2, ju1:j2)
real*8  :: ai              (k1-1:k2)
real*8  :: bi              (k1-1:k2)
real*8  :: humidity        (ii:i2, ju1:j2, k1:k2)
real*8  :: pctm1           (ilo:ih1, julo:jhi)
real*8 , optional :: cldmas0 (ii:i2, ju1:j2)
real*8 , optional :: press3c (ilo:ih1, julo:jhi, k1:k2)
real*8 , optional :: press3e (ilo:ih1, julo:jhi, k1-1:k2)
integer, optional :: cmi_flags (ii:i2, ju1:j2)
real*8 , optional :: dtrn   (ii:i2, ju1:j2, k1:k2)
real*8 , optional :: flashrate (ii:i2, ju1:j2)
real*8 , optional :: lightning_no (ii:i2, ju1:j2, k1:k2)

```

B Legacy Code: Variables used in the Emission Operator

NAME	DESCRIPTION	UNITS	INTENT	OWNERSHIP
chem_opt	Chemistry option		IN	Chemistry

trans_opt	Transport option	IN	Advection	
do_drydep	Do dry deposition?	IN	Deposition	
press3c	Atmospheric pressure at the center of each grid box [mb]	IN	local/shared	
press3e	Atmospheric pressure at the edge of each grid box [mb]	IN	local/shared	
cldmas0	Normalized cldmas at desired level	IN	local	
met_opt	MetFields option	IN	MetFields	
cmi_flags	Array of flags that indicate continental, marine, or ice	IN	MetFields	
dtrn	Detrainment rate [DAO:kg/m^2*s, NCAR:s^-1]	IN	MetFields	
lwi_flags	Array of flags that indicate land, water, or ice	IN	MetFields	
emiss_opt	Emission option	IN	Emission	
lightning_opt	Lightning option	IN	Emission	
emiss_aero_opt	Sulfur aerosol emiss option	IN	Emission	
emiss_dust_opt	Sulfur dust emiss option	IN	Emission	
emiss_map	mapping of emission number to const species #	IN	Emission	
emiss_map_dust	mapping of dust emiss number to const species #	IN	Emission	
emiss_map_aero	mapping of aerosol emiss number to const species #	IN	Emission	
isop_scale	Array of monthly isoprene scaling coefficients	IN	Emission	
do_gcr	Do Galactic Cosmic Ray	IN	Emission	
num_emiss	Number of emissions	IN	Emission	
emiss_timpqr	Emission time per year	IN	Emission	
ndust	Number of dust	IN	Emission	
nst_dust	starting index for dust	IN	Emission	
nt_dust	ending index for dust	IN	Emission	
naero	number of aerosols	IN	Emission	
do_semiss_inchem	Do surface emissions inside chemistry solver?	IN	Emission	
isop_scale	array of monthly isoprene scaling coefficients	IN	Emission	
desired_g_N_prod_rate	Lightning option	IN	Emission	
gcr_infile_name	Galactic Cosmic Ray input file name	IN	Emission	
fertscal_infile_name	fertilizer scale input file name	IN	Emission	
lai_infile_name	leaf area index input file name	IN	Emission	
light_infile_name	light input file name	IN	Emission	
precip_infile_name	precipitation input file name	IN	Emission	
soil_infile_name	soil type input file name	IN	Emission	
veg_infile_name	vegetation type input file name	IN	Emission	
isopconv_infile_name	isoprene convert input file name	IN	Emission	
monotconv_infile_name	monoterpene convert input file name	IN	Emission	
emiss_dust_t	Array of dust emissions [kg/s]	IN	Emission	
emiss_dust	Array of dust emissions for 6 hours [kg/s]	OUT	Emission	
emiss_aero	Array of aerosol emissions [kg/s]	OUT	Emission	
flashrate	Flash rate (flashes per 4x5 box per s)	OUT	Emission	
lightning_no	3d array of pnox production in kg./sec	OUT	Emission	
emiss_isop	Isoprene emissions [kg/s]	OUT	Emission	
emiss_monot	Monoterpene emissions [kg/s]	OUT	Emission	
emiss_nox	NOx emissions [kg/s]	OUT	Emission	
emiss	Array of emissions [kg/s]	INOUT	Emission	
surf_emiss_out	Surface emissions to be written to output [kg/m^2/time]	INOUT	Emission/Diagn.	
emiss_3d_out	Surface emissions to be written to output [kg/m^2/time]	INOUT	Emission/Diagn.	
surf_emiss_out2	Surface emissions to be written to output [kg/m^2/time]	INOUT	Emission/Diagn.	
num_species	Number of species	IN	setkin/shared	
iisoprene_num	const array index for isoprene (C5H8)	IN	setkin/shared	
ino_num	const array index for NO	IN	setkin/shared	
ico_num	const array index for CO	IN	setkin/shared	
lpropene_num	const array index for propene (C3H6)	IN	setkin/shared	
mw	Array of species' molecular weights [g/mol]	IN	setkin/shared	
pt	pressure = (am * pt) + (bm * psx)	[mb]	IN	Shared
tdt	model time step	[s]	IN	Shared
ndt	model time step	[s]	IN	Shared
nymd	current year/month/day (YYMMDD)		IN	Shared
nhms	current hour/min/sec (HHMMSS)		IN	Shared
num_time_steps	number of time steps		IN	Shared
ai	Pressure = (ai * pt) + (bi * psx), ai at zone interface	IN	Shared	
bi	Pressure = (ai * pt) + (bi * psx), bi at zone interface	IN	Shared	
latdeg	Latitude [deg]	IN	Shared	
londeg	Longitude [deg]	IN	Shared	
mcor	Area of grid box [m^2]	IN	Shared	
mass	Total mass of the atmosphere within each grid box [kg]	IN	Shared	
radswg	Net downward shortwave radiation at ground [W/m^2]	IN	MetFields	
surf_air_temp	Surface air temperature [degK]	IN	MetFields	
surf_rough	Surface roughness [m]	IN	MetFields	
con_precip	Convective precipitation [mm/day]	IN	MetFields	
tot_precip	Total precipitation [mm/day]	IN	MetFields	
ustar	Ustar [m/s]	IN	MetFields	
max_cloud	Maximum overlap cloud fraction for LW	IN	MetFields	
ran_cloud	Random overlap cloud fraction for LW	IN	MetFields	
kel	Temperature [degK]	IN	MetFields	
pbl	Boundary layer height [m]	IN	MetFields	
humidity	Specific humidity [g/kg]	IN	MetFields	
pctm1	Surface pressure at t1 [mb]	IN	MetFields	
const	Species concentration, known at zone centers [mixing ratio]	INOUT	ALL	
i1, i2	Domain decomposition	IN	Shared	
ju1, j2	Domain decomposition	IN	Shared	
k1, k2	Domain decomposition	IN	Shared	
ilo, ihi	Domain decomposition	IN	Shared	
julo, jhi	Domain decomposition	IN	Shared	

i1_gl, i2_gl	Domain decomposition	IN	Shared
j1_gl, j2_gl	Domain decomposition	IN	Shared

C Emission Derived Type and Member Variable Manipulations

```

module Emission_Mod

use m_species_parameters, only : MAX_NUM_CONST
use m_SetSpeciesNumbers , only : num_species

implicit none

private
public :: Allocate_emiss      , Allocate_emiss_isop      , Allocate_emiss_monot,
          Allocate_emiss_nox     , Allocate_lightning_no     , Allocate_flashrate
public :: Allocate_surf_emiss_out, Allocate_surf_emiss_out2, Allocate_emiss_3d_out
public :: Allocate_emiss_dust   , Allocate_emiss_dust_t   , Allocate_emiss_aero
public :: Allocate_emiss_aero_t

public :: Set_emiss           , Set_emiss_isop           , Set_emiss_monot , Set_emiss_nox
public :: Set_surf_emiss_out , Set_surf_emiss_out2 , Set_emiss_3d_out
public :: Set_lightning_no    , Set_flashrate           , Set_emiss_dust , Set_emiss_dust_t
public :: Set_emiss_aero     , Set_emiss_aero_t

public :: Get_emiss           , Get_emiss_isop           , Get_emiss_monot
public :: Get_emiss_nox       , Get_surf_emiss_out      , Get_surf_emiss_out2
public :: Get_emiss_3d_out    , Get_lightning_no        , Get_flashrate
public :: Get_emiss_dust      , Get_emiss_dust_t        , Get_emiss_aero
public :: Get_emiss_aero_t    , Get_do_semiss_inchem   , Get_desired_g_N_prod_rate
public :: Get_fertscal_infile_name, Get_light_infile_name , Get_lai_infile_name
public :: Get_precip_infile_name, Get_soil_infile_name   , Get_veg_infile_name
public :: Get_isopconv_infile_name, Get_monotconv_infile_name
public :: Get_lightning_opt   , Get_emiss_aero_opt      , Get_emiss_dust_opt
public :: Get_num_emiss       , Get_ndust , Get_naero    , Get_nst_dust
public :: Get_nt_dust         , Get_emiss_map_dust     , Get_emiss_map_aero
public :: Get_emiss_map_aero  , Get_isop_scale          , Get_do_gcr

public :: ReadEmissionNamelist

public :: Emission

type Emission
  private
  real*8      :: isop_scale (12)
  integer      :: i_no_lgt
  real*8      :: desired_g_N_prod_rate
  integer      :: semiss_inchem_flag
  integer      :: emiss_opt      , emiss_in_opt      , emiss_aero_opt
  integer      :: emiss_dust_opt , lightning_opt     , emiss_conv_flag
  real*8      :: emiss_conv_fac , emiss_init_val
  integer      :: num_emiss      , ndust            , naero
  integer      :: nst_dust       , nt_dust           , emiss_timpqr
  logical      :: do_gcr         , do_semiss_inchem
  character (len=128) :: emiss_dust_infile_name, emiss_infile_name
  character (len=128) :: emiss_aero_infile_name, gcr_infile_name
  character (len=128) :: fertscal_infile_name , lai_infile_name
  character (len=128) :: light_infile_name  , precip_infile_name
  character (len=128) :: soil_infile_name   , veg_infile_name
  character (len=128) :: isopconv_infile_name, monotconv_infile_name
  character (len=32)  :: emiss_var_name
  integer       :: emiss_map(MAX_NUM_CONST)
  integer       :: emiss_map_aero(MAX_NUM_CONST)
  integer       :: emiss_map_dust(MAX_NUM_CONST)
  real*8 , pointer :: emiss      (:,:,:,:,:) => null()
  real*8 , pointer :: emiss_isop  (:,:,:) => null()
  real*8 , pointer :: emiss_monot (:,:,:) => null()
  real*8 , pointer :: emiss_nox   (:,:,:) => null()
  real*8 , pointer :: flashrate   (:,:,:) => null()
  real*8 , pointer :: lightning_no (:,:,:,:) => null()
  real*8 , pointer :: emiss_dust  (:,:,:,:) => null()
  real*8 , pointer :: emiss_aero   (:,:,:,:) => null()
  real*8 , pointer :: emiss_dust_t (:,:,:,:,:) => null()
  real*8 , pointer :: emiss_aero_t (:,:,:,:,:) => null()
  real*8 , pointer :: surf_emiss_out (:,:,:,:) => null()
  real*8 , pointer :: surf_emiss_out2(:,:,:,:) => null()
  real*8 , pointer :: emiss_3d_out  (:,:,:,:,:) => null()

end type Emission

!-----
CONTAINS
!-----
subroutine Allocate_emiss (self, GMIdims)
  use m_dimensions_vars , only : GMIDimensions
  type (GMIDimensions) , intent(in)    :: GMIdims
  type (Emission)      , intent(inout) :: self

```

```

Allocate(self%emiss(GMIdims%ii1:GMIdims%ii2, GMIdims%ju1:GMIdims%ju2, GMIdims%ki1:GMIdims%ki2, self%num_emiss, self%emiss_timpqr))
self%emiss = 0.0d0
return
end subroutine Allocate_emiss
!-----
subroutine Get_emiss (self, emiss)
real*8 , intent(out) :: emiss (:,:,:,:)
type (Emission) , intent(in) :: self
emiss(:,:,:,:) = self%emiss(:,:,:,:)
return
end subroutine Get_emiss
!-----
subroutine Set_emiss (self, emiss)
real*8 , intent(in) :: emiss (:,:,:,:)
type (Emission) , intent(inout) :: self
self%emiss(:,:,:,:) = emiss(:,:,:,:)
return
end subroutine Set_emiss
!-----
subroutine Get_veg_infile_name (self, veg_infile_name)
character (len=128), intent(out) :: veg_infile_name
type (Emission) , intent(in) :: self
veg_infile_name = self%veg_infile_name
return
end subroutine Get_veg_infile_name
!-----
subroutine Get_isop_scale (self, isop_scale)
real*8 , intent(out) :: isop_scale(:)
type (Emission), intent(in) :: self
isop_scale(:) = self%isop_scale(:)
return
end subroutine Get_isop_scale
!-----
.
.
.
!-----
end module Emission_Mod

```

D Code for the RunEmission Routine

```

subroutine RunEmission (press3e, press3c,
& chem_opt, met_opt, trans_opt,
& pr_surf_emiss, pr_emiss_3d, do_drydep,
& t_emission, t_MassPressure, t_MetFields,
& t_SpeciesConcentration, GMStepping, GMIdims)

use m_SetSpeciesNumbers , only : num_species
use m_SetSpeciesIndices , only : iisoprene_num, ino_num, ico_num, ipropene_num
use m_SetMolecularWeight , only : mw

use m_Emission_Update , only : Update_Emiss

use Emission_Mod , only : Get_emiss_opt , Get_emiss_in_opt
use Emission_Mod , only : Get_lightning_opt , Get_isop_scale
use Emission_Mod , only : Get_veg_infile_name , Get_lai_infile_name
use Emission_Mod , only : Get_isopconv_infile_name , Get_monotconv_infile_name
use Emission_Mod , only : Get_fertscal_infile_name , Get_precip_infile_name
use Emission_Mod , only : Get_soil_infile_name , Get_light_infile_name
use Emission_Mod , only : Get_surf_emiss_out , Get_surf_emiss_out2
use Emission_Mod , only : Get_emiss_3d_out
use Emission_Mod , only : Get_emiss , Get_emiss_map
use Emission_Mod , only : Get_emiss_dust_t , Get_emiss_dust
use Emission_Mod , only : Get_emiss_aero_t , Get_emiss_aero
use Emission_Mod , only : Get_do_gcr , et_do_semiss_inchem
use Emission_Mod , only : Get_emiss_map_dust , Get_emiss_map_aero
use Emission_Mod , only : Get_emiss_aero_opt , Get_emiss_dust_opt
use Emission_Mod , only : Get_num_emiss , Get_emiss_timpqr
use Emission_Mod , only : Get_naero , Get_ndust
use Emission_Mod , only : Get_nst_dust , Get_nt_dust
use Emission_Mod , only : Get_desired_g_N_prod_rate , Get_flashrate
use Emission_Mod , only : Get_lightning_no

use Emission_Mod , only : Set_emiss , Set_emiss_nox
use Emission_Mod , only : Set_emiss_isop , Set_emiss_isop
use Emission_Mod , only : Set_emiss_monot , Set_emiss_dust
use Emission_Mod , only : Set_emiss_aero , Set_flashrate
use Emission_Mod , only : Set_lightning_no , Set_emiss_3d_out
use Emission_Mod , only : Set_surf_emiss_out , Set_surf_emiss_out2

use m_MetFields , only : MetFields
use m_MetFields , only : Get_kel , Get_humidity
use m_MetFields , only : Get_max_cloud , Get_ran_cloud
use m_MetFields , only : Get_pbl , Get_cmf
use m_MetFields , only : Get_pctm1 , Get_dtrn
use m_MetFields , only : Get_lwi_flags , Get_cmi_flags
use m_MetFields , only : Get_con_precip , Get_tot_precip
use m_MetFields , only : Get_surf_air_temp , Get_surf_rough

```

```

use m_MetFields      , only : Get_ustar           , Get_zmmu
use m_MetFields      , only : Get_metdata_name_org , Get_radswg

use m_dimensions_vars , only : GMI_Dimensions
use m_dimensions_vars , only : Get_GMI_Dimensions_loc   , Get_GMI_Dimensions_glo
use m_dimensions_vars , only : Get_GMI_Dimensions_loc_ghost

use m_ModelSteppingVars , only : ModelStepping
use m_ModelSteppingVars , only : Get_tdt            , Get_ndt
use m_ModelSteppingVars , only : Get_pt             , Get_num_time_steps
use m_ModelSteppingVars , only : Get_nhms           , Get_nynd

use m_MassPressure    , only : MassPressure
use m_MassPressure    , only : Get_ai              , Get_bi
use m_MassPressure    , only : Get_mass             , Get_mcpr
use m_MassPressure    , only : Get_latdeg          , Get_londeg

use m_SpeciesConcentration, only : SpeciesConcentration
use m_SpeciesConcentration, only : Get_const          , Set_const
!

! Arguments of the routine
!
type (Emission      ), intent(inout) :: t_emission
type (MassPressure   ), intent(in ) :: t_MassPressure
type (MetFields      ), intent(inout) :: t_MetFields
type (GMI_Dimensions ), intent(in ) :: GMIdims
type (ModelStepping  ), intent(in ) :: GMIstepping
type (SpeciesConcentration), intent(inout) :: t_SpeciesConcentration

integer, intent(in) :: chem_opt, met_opt, trans_opt
logical, intent(in) :: pr_surf_emiss, pr_emiss_3d
logical, intent(in) :: do_drydep
real*8 , intent(in) :: press3e(GMIdims%ilo:GMIdims%jhi, GMIdims%julo:GMIdims%jhi, GMIdims%k1-1:GMIdims%k2)
real*8 , intent(in) :: press3c(GMIdims%ilo:GMIdims%jhi, GMIdims%julo:GMIdims%jhi, GMIdims%k1:GMIdims%k2)
!

! Variables to be set by derived types
!
real*8 :: tdt, pt
integer :: ndt
integer :: nynd, nhms, num_time_steps

real*8 , allocatable :: const(:,:,:,:)

real*8 , allocatable :: ai      (:)
real*8 , allocatable :: bi      (:)
real*8 , allocatable :: latdeg (:)
real*8 , allocatable :: londeg (:)
real*8 , allocatable :: mcpr   (:,:)
real*8 , allocatable :: mass   (:,:,:)

real*8 , allocatable :: kel     (:,:,:,:)
real*8 , allocatable :: pbl     (:,:)
real*8 , allocatable :: cmf     (:,:,:,:)
real*8 , allocatable :: zmmu   (:,:,:,:)
real*8 , allocatable :: dtrn   (:,:,:,:)
real*8 , allocatable :: pctm1  (:,:)
real*8 , allocatable :: ustар  (:,:)
real*8 , allocatable :: radswg (:,:)
real*8 , allocatable :: humidity(:,:,:,:)
integer, allocatable :: lwi_flags(:,:)
integer, allocatable :: cmi_flags(:,:)
real*8 , allocatable :: max_cloud(:,:,:,:)
real*8 , allocatable :: ran_cloud(:,:,:,:)
real*8 , allocatable :: tot_precip(:,:)
real*8 , allocatable :: con_precip(:,:)
real*8 , allocatable :: surf_rough(:,:)
real*8 , allocatable :: surf_air_temp(:,:)

character (len=16) :: metdata_name_org

integer :: emiss_opt, emiss_in_opt, emiss_aero_opt, emiss_dust_opt, lightning_opt
integer :: emiss_timpqr, naero, ndust, nt_dust, nt_dust, num_emiss
integer :: emiss_map      (num_species)
integer :: emiss_map_dust (num_species)
integer :: emiss_map_aero (num_species)
real*8 :: isop_scale(12)
real*8 :: desired_g_N_prod_rate
logical :: do_gcr, do_semiss_inchem
real*8 , allocatable :: lightning_no  (:,:,:,:)
real*8 , allocatable :: flashrate   (:,:)
real*8 , allocatable :: emiss_dust   (:,:,:,:)
real*8 , allocatable :: emiss_aero   (:,:,:,:)
real*8 , allocatable :: surf_emiss_out(:,:,:,:)
real*8 , allocatable :: surf_emiss_out2(:,:,:,:)
real*8 , allocatable :: emiss_3d_out (:,:,:,:,:)
real*8 , allocatable :: emiss_dust_t (:,:,:,:,:)
real*8 , allocatable :: emiss_aero_t (:,:,:,:,:)
real*8 , allocatable :: emiss        (:,:,:,:,:,:)
real*8 , allocatable :: emiss_nox   (:,:)
real*8 , allocatable :: emiss_isop   (:,:)
real*8 , allocatable :: emiss_monot (:,:)

```

```

character (len=128) :: fertscal_infile_name
character (len=128) :: lai_infile_name
character (len=128) :: light_infile_name
character (len=128) :: precip_infile_name
character (len=128) :: soil_infile_name
character (len=128) :: veg_infile_name
character (len=128) :: isopconv_infile_name
character (len=128) :: monotconv_infile_name

integer :: i1, i2, ju1, jv1, j2, k1, k2
integer :: ilo, ihi, jul0, jvlo, jhi
integer :: i1_gl, i2_gl, ju1_gl, jv1_gl, j2_gl, k1_gl, k2_gl
!
! Local Variables
!
real*8 , allocatable :: cldmas0(:,:)

!!!!!!!!!!!
! Code Begin
!!!!!!!!!!!

call Get_do_gcr          (t_emission, do_gcr           )
call Get_emiss_opt        (t_emission, emiss_opt       )
if ((emiss_opt /= 0) .or. do_drydep .or. do_gcr) then
    ! Getting variables from the derived types.
    ! The variables are the arguments of Update_Emiss that are IN or INOUT.

call Get_GMI_Dimensions_loc      (GMIDims, i1, i2, ju1, jv1, j2, k1, k2 )
call Get_GMI_Dimensions_loc_ghost (GMIDims, ilo, ihi, jul0, jvlo, jhi   )
call Get_GMI_Dimensions_glo      (GMIDims, i1_gl, i2_gl, ju1_gl, jv1_gl, j2_gl, k1_gl, k2_gl)

call Get_emiss_in_opt          (t_emission, emiss_in_opt     )
call Get_lightning_opt         (t_emission, lightning_opt   )
call Get_emiss_aero_opt        (t_emission, emiss_aero_opt  )
call Get_emiss_dust_opt        (t_emission, emiss_dust_opt  )
call Get_emiss_timpqr          (t_emission, emiss_timpqr    )
call Get_num_emiss             (t_emission, num_emiss      )
call Get_ndust                 (t_emission, ndust          )
call Get_naero                 (t_emission, naero          )
call Get_nst_dust              (t_emission, nst_dust        )
call Get_nt_dust               (t_emission, nt_dust         )
call Get_emiss_map              (t_emission, emiss_map      )
call Get_emiss_map_dust        (t_emission, emiss_map_dust  )
call Get_emiss_map_aero        (t_emission, emiss_map_aero  )
call Get_isop_scale            (t_emission, isop_scale      )
call Get_do_semiss_inchem      (t_emission, do_semiss_inchem)
call Get_desired_g_N_prod_rate (t_emission, desired_g_N_prod_rate)

call Get_fertscal_infile_name  (t_emission, fertscal_infile_name )
call Get_light_infile_name     (t_emission, light_infile_name  )
call Get_lai_infile_name       (t_emission, lai_infile_name   )
call Get_precip_infile_name   (t_emission, precip_infile_name)
call Get_soil_infile_name     (t_emission, soil_infile_name  )
call Get_veg_infile_name      (t_emission, veg_infile_name  )
call Get_isopconv_infile_name (t_emission, isopconv_infile_name)
call Get_monotconv_infile_name(t_emission, monotconv_infile_name)

if (emiss_in_opt /= 0) then
    Allocate (emiss (i1:i2, ju1:j2, k1:k2, num_emiss, emiss_timpqr))
    call Get_emiss          (t_emission, emiss           )
end if

if (emiss_opt == 2) then
    Allocate (emiss_monot (i1:i2, ju1:j2))
    Allocate (emiss_isop  (i1:i2, ju1:j2))
    Allocate (emiss_nox   (i1:i2, ju1:j2))
end if

if (pr_emiss_3d) then
    Allocate (emiss_3d_out(i1:i2, ju1:j2, k1:k2, 1:num_species))
    call Get_emiss_3d_out (t_emission, emiss_3d_out)
end if

if (pr_surf_emiss) then
    Allocate (surf_emiss_out(i1:i2, ju1:j2, 1:num_species))
    Allocate (surf_emiss_out2(i1:i2, ju1:j2, 1:4))
    call Get_surf_emiss_out2 (t_emission, surf_emiss_out2)
    call Get_surf_emiss_out (t_emission, surf_emiss_out )
end if

if (emiss_aero_opt /= 0) then
    Allocate (emiss_aero (i1:i2, ju1:j2, 1:naero))
    Allocate (emiss_aero_t(i1:i2, ju1:j2, 1:naero, emiss_timpqr))
    call Get_emiss_aero_t (t_emission, emiss_aero_t )
end if

if (emiss_dust_opt /= 0) then
    Allocate (emiss_dust (i1:i2, ju1:j2, 1:ndust))
    Allocate (emiss_dust_t(i1:i2, ju1:j2, 1:ndust, nst_dust:nst_dust+nt_dust-1))
    call Get_emiss_dust_t (t_emission, emiss_dust_t )
end if

```

```

if (lightning_opt == 1) then
  Allocate ( flashrate(i1:i2, ju1:j2))
  Allocate ( lightning_no(i1:i2, ju1:j2, k1:k2))
  call Get_flashrate   (t_emission, flashrate      )
  call Get_lightning_no (t_emission, lightning_no  )
end if

allocate(const(i1:i2, ju1:j2, k1:k2, num_species))
call Get_const           (t_SpeciesConcentration, const)

allocate(ai      (k1-1:k2))
allocate(bi      (k1-1:k2))
allocate(mcor    (i1:i2, ju1:j2))
allocate(mass   (i1:i2, ju1:j2, k1:k2))
allocate(latdeg(ju1_g1:j2,g1))
allocate(londeg(i1_g1:i2_g1))
call Get_ai        (t_MassPressure, ai      )
call Get_bi        (t_MassPressure, bi      )
call Get_mass     (t_MassPressure, mass    )
call Get_mcor     (t_MassPressure, mcor    )
call Get_latdeg   (t_MassPressure, latdeg  )
call Get_londeg   (t_MassPressure, londeg  )

call Get_tdt       (GMStepping, tdt      )
call Get_ndt       (GMStepping, ndt      )
call Get_pt        (GMStepping, pt       )
call Get_nymd     (GMStepping, nymd    )
call Get_nhms     (GMStepping, nhms    )
call Get_num_time_steps (GMStepping, num_time_steps)

allocate(kel      (ilo:ihi, jul0:jhi, k1:k2))
allocate(pbl      (i1:i2, ju1:j2))
allocate(cmf      (i1:i2, ju1:j2, k1:k2))
allocate(pctm1   (ilo:ihi, jul0:jhi))
allocate(ustar    (i1:i2, ju1:j2))
allocate(radswg   (i1:i2, ju1:j2))
allocate(humidity (i1:i2, ju1:j2, k1:k2))
allocate(lwi_flags (i1:i2, ju1:j2))
allocate(max_cloud (i1:i2, ju1:j2, k1:k2))
allocate(ran_cloud (i1:i2, ju1:j2, k1:k2))
allocate(surf_rough (i1:i2, ju1:j2))
allocate(tot_precip (i1:i2, ju1:j2))
allocate(con_precip (i1:i2, ju1:j2))
allocate(surf_air_temp(i1:i2, ju1:j2))

call Get_metadata_name_org (t_MetFields, metadata_name_org)
call Get_kel    (t_MetFields, kel      )
call Get_humidity (t_MetFields, humidity )
call Get_max_cloud (t_MetFields, max_cloud )
call Get_ran_cloud (t_MetFields, ran_cloud )
call Get_con_precip (t_MetFields, con_precip)
call Get_tot_precip (t_MetFields, tot_precip)
call Get_surf_air_temp (t_MetFields, surf_air_temp)
call Get_surf_rough (t_MetFields, surf_rough )
call Get_ustar    (t_MetFields, ustar    )
call Get_pbl     (t_MetFields, pbl      )
call Get_radswg  (t_MetFields, radswg  )
call Get_cmf     (t_MetFields, cmf      )
call Get_pctm1   (t_MetFields, pctm1   )
call Get_lwi_flags (t_MetFields, lwi_flags )

if (lightning_opt == 1) then
  allocate(dtrn    (i1:i2, ju1:j2, k1:k2))
  allocate(cmi_flags (i1:i2, ju1:j2))
  call Get_cmi_flags (t_MetFields, cmi_flags)
  call Get_dtrn   (t_MetFields, dtrn    )
endif
!
! Beginning of the legacy code section
!
if ((emiss_opt /= 0) .or. do_drydep .or. do_gcr) then

  if (lightning_opt == 1) then
    allocate(cldmas0(i1:i2, ju1:j2))
    if (metadata_name_org(1:3) == 'DAO') then
      cldmas0 = cmf(:,12)
    else if(metadata_name_org(1:4) == 'NCAR') then
      cldmas0 = cmf(:,10)
    else if(metadata_name_org(1:4) == 'GISS') then
      cldmas0 = cmf(:,8)
    else if(metadata_name_org(1:4) == 'GMAO') then
      call Get_zmmu (t_MetFields, zmmu )
      cldmas0 = zmmu(:,9)
    else
      write (6,*) 'Lightning parameterization does not exist for this model'
    endif

    call Update_Emiss  &
    & (lwi_flags, latdeg, londeg, mcor, emiss_isop, emiss_monot,  &
    & emiss_nox, radswg, surf_air_temp, surf_rough, con_precip,  &
    & tot_precip, ustar, mass, max_cloud, ran_cloud, kel,   &

```

```

&      surf_emiss_out, surf_emiss_out2, emiss_3d_out,   &
&      const, emiss, emiss_dust_t, emiss_dust, emiss_aero_t,   &
&      emiss_aero, pbl, ai, bi, humidity, pctm1,   &
&      metadata_name_org, veg_infile_name, lai_infile_name,   &
&      isopconv_infile_name, monottconv_infile_name, fertscal_infile_name,   &
&      precip_infile_name, soil_infile_name, light_infile_name,   &
&      pr_surf_emiss, pr_emiss_3d,   &
&      desired_g_N_prod_rate, isop_scale, do_semiss_inchem, do_gcr, do_drydep,   &
&      emiss_map, emiss_map_dust, emiss_map_aero, mw,   &
&      tdt, ndt, pt, nymd, nhms, num_time_steps,   &
&      iisoprene_num, ino_num, ico_num, ipropene_num,   &
&      lightning_opt, emiss_opt, emiss_aero_opt, emiss_dust_opt, met_opt,   &
&      chem_opt, trans_opt,   &
&      num_species, num_emiss, emiss_timpqr, naero, ndust, nst_dust, nt_dust,   &
&      GMIdims,   &
&      cldmas0, press3c, press3e, cmi_flags, dtrn, flashrate, lightning_no)
else
call Update_Emiss   &
  (lwi_flags, latdeg, londeg, mc当地, emiss_isop, emiss_monot,   &
  emiss_nox, radswg, surf_air_temp, surf_rough, con_precip,   &
  tot_precip, ustar, mass, max_cloud, ran_cloud, kel,   &
  surf_emiss_out, surf_emiss_out2, emiss_3d_out,   &
  const, emiss, emiss_dust_t, emiss_dust, emiss_aero_t,   &
  emiss_aero, pbl, ai, bi, humidity, pctm1,   &
  metadata_name_org, veg_infile_name, lai_infile_name,   &
  isopconv_infile_name, monottconv_infile_name, fertscal_infile_name,   &
  precip_infile_name, soil_infile_name, light_infile_name,   &
  pr_surf_emiss, pr_emiss_3d,   &
  desired_g_N_prod_rate, isop_scale, do_semiss_inchem, do_gcr, do_drydep,   &
  emiss_map, emiss_map_dust, emiss_map_aero, mw,   &
  tdt, ndt, pt, nymd, nhms, num_time_steps,   &
  iisoprene_num, ino_num, ico_num, ipropene_num,   &
  lightning_opt, emiss_opt, emiss_aero_opt, emiss_dust_opt, met_opt,   &
  chem_opt, trans_opt,   &
  num_species, num_emiss, emiss_timpqr, naero, ndust, nst_dust, nt_dust,   &
  GMIdims)

endif
endif
!
! Ending of the legacy code section
!

! Passing the updated variables to the derived types before exiting the routine.
! The variables are the arguments of Update_Emiss that are INOUT or OUT.

call Set_const      (t_SpeciesConcentration, const)

if (emiss_in_opt /= 0) then
  call Set_emiss  (t_emission, emiss )
end if

if (emiss_opt == 2) then
  call Set_emiss_monot  (t_emission, emiss_monot )
  call Set_emiss_isop   (t_emission, emiss_isop )
  call Set_emiss_nox   (t_emission, emiss_nox )
end if

if (emiss_aero_opt /= 0) then
  call Set_emiss_aero  (t_emission, emiss_aero )
endif

if (emiss_dust_opt /= 0) then
  call Set_emiss_dust  (t_emission, emiss_dust )
endif

if (lightning_opt == 1) then
  call Set_lightning_no (t_emission, lightning_no )
  call Set_flashrate   (t_emission, flashrate     )
endif

if (pr_surf_emiss) then
  call Set_surf_emiss_out (t_emission, surf_emiss_out )
  call Set_surf_emiss_out2(t_emission, surf_emiss_out2)
end if

if (pr_emiss_3d) then
  call Get_emiss_3d_out (t_emission, emiss_3d_out)
end if

end if

return

end subroutine RunEmission

```